Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

Технології розроблення програмного забезпечення

Тема: Патерни проектування

Виконав
студент групи ІА-34:
Жительний В.В.

Перевірив:
Мягкий М.Ю.

Київ 2025

Мета: Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» та навчитися застосовувати їх в реалізації програмної системи.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Система має складну внутрішню структуру з багатьма сервісами. Щоб не перевантажувати контролери залежностями та бізнес-логікою, було впроваджено Фасад. Він надає простий, уніфікований інтерфейс для контролерів, приховуючи складність взаємодії між підсистемами, наприклад, при створенні станції Фасад одночасно зберігає її в БД і запускає процес стрімінгу.

Графічне представлення структури реалізації даного шаблону наведено на діаграмі класів

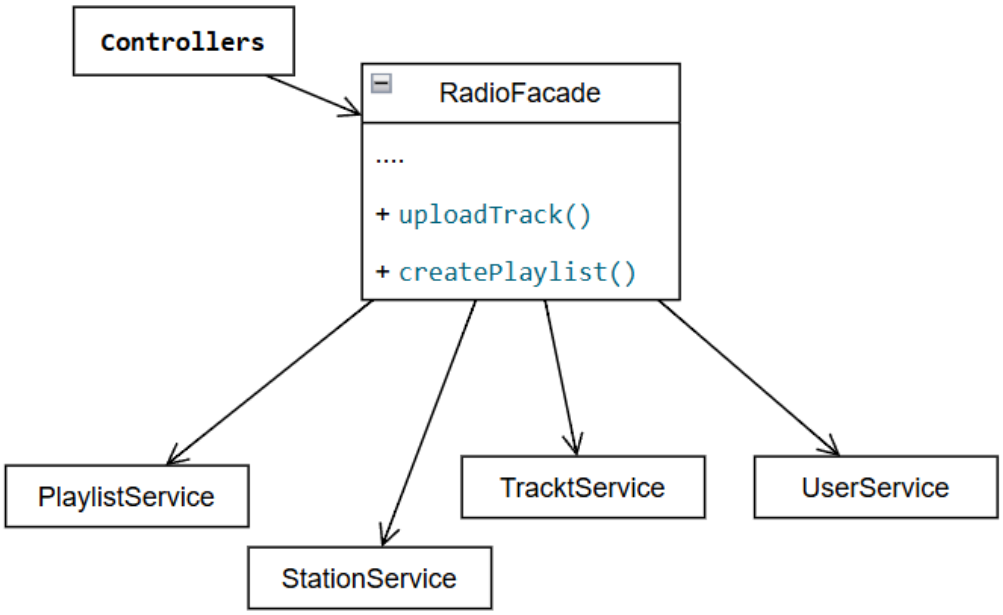


Рис. 1 Діаграма класів для шаблону Фасад

Коротко описуючи діаграму: Центральним елементом є клас RadioFacade, який виступає єдиною точкою входу для зовнішніх запитів від контролерів. Він приховує складність внутрішньої архітектури, взаємодіючи з підсистемами PlaylistService, StationService, TrackService, UserService та іншими. Фасад надає спрощені методи, такі як uploadTrack() та createPlaylist(), делегуючи виконання конкретних бізнес-операцій відповідним сервісам, що дозволяє зменшити зв'язність між клієнтським кодом і складною логікою системи.

Програмна реалізація

```
package com.zhytelnyi.online_radio_server.service.facade;

import com.zhytelnyi.online_radio_server.model.*;
import com.zhytelnyi.online_radio_server.repository.ConnectionLogRepository;
import com.zhytelnyi.online_radio_server.repository.ListenLogRepository;
import com.zhytelnyi.online_radio_server.service.core.RadioServer;
import com.zhytelnyi.online_radio_server.service.domain.*;
import com.zhytelnyi.online_radio_server.service.adapter.IReportExporter;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.util.List;

@Service
public class RadioFacade {

    private final TrackService trackService;
    private final PlaylistService playlistService;
    private final StationService stationService;
    private final UserService userService;
    private final RadioServer radioServer;
    private final IReportExporter reportExporter;
    private final ListenLogRepository listenLogRepository;
    private final ConnectionLogRepository connectionLogRepository;

    public RadioFacade(TrackService trackService,
                       PlaylistService playlistService,
                       StationService stationService,
                       UserService userService,
                       RadioServer radioServer,
                       @Qualifier("jsonExporter") IReportExporter
reportExporter, ListenLogRepository listenLogRepository, ConnectionLogRepository
connectionLogRepository) {
        this.trackService = trackService;
        this.playlistService = playlistService;
        this.stationService = stationService;
        this.userService = userService;
        this.radioServer = radioServer;
        this.reportExporter = reportExporter;
        this.listenLogRepository = listenLogRepository;
        this.connectionLogRepository = connectionLogRepository;
    }

    // === TRACKS ===
    public void uploadTrack(String title, String artist, MultipartFile file) {
        trackService.uploadTrack(title, artist, file);
    }
    public List<Track> getAllTracks() { return trackService.findAll(); }
    public void deleteTrack(Long id) { trackService.delete(id); }

    // === PLAYLISTS ===
    public void createPlaylist(String name, List<Long> trackIds) {
```

```
        List<Track> tracks = (trackIds != null) ?
trackService.findAllByIds(trackIds) : null;
        playlistService.create(name, tracks);
    }

    public void updatePlaylist(Long id, String newName, List<Long> trackIds) {
        List<Track> tracks = (trackIds != null) ?
trackService.findAllByIds(trackIds) : null;
        playlistService.update(id, newName, tracks);

        Playlist updatedPlaylist = playlistService.findById(id);
        List<Station> affectedStations =
stationService.findAllByPlaylist(updatedPlaylist);

        for (Station station : affectedStations) {
            try {
                radioServer.restartChunking(station);
            } catch (Exception e) {
                System.err.println("Failed to restart stream for station " +
station.getName());
            }
        }
    }
    public Playlist getPlaylistById(Long id) { return
playlistService.findById(id); }
    public List<Playlist> getAllPlaylists() { return playlistService.findAll();
}
    public void deletePlaylist(Long id) { playlistService.delete(id); }

    // === STATIONS ===
    public void createStation(String name, int bitrate, List<Long> playlistIds)
{
        List<Playlist> playlists = (playlistIds != null) ?
playlistService.findAllByIds(playlistIds) : null;

        Station savedStation = stationService.create(name, bitrate, playlists);

        try {
            radioServer.startChunking(savedStation);
        } catch (Exception e) {
            System.err.println("Warning: Failed to start stream immediately: " +
e.getMessage());
        }
    }

    public List<Station> getAllStations() { return stationService.findAll(); }
    public void deleteStation(Long id) { stationService.delete(id); }

    public String getStationReport(Long id) {
        Station station = stationService.findById(id);
        if (station == null) return "{\"error\": \"Not Found\"}";
        return reportExporter.export(station);
    }

    // === FAVORITES ===
    public String addStationToFavorites(String username, Long stationId) {
        return userService.addFavorite(username, stationId);
    }
    public List<Station> getUserFavoriteStations(String username) {
        return userService.getFavorites(username);
    }

    // === STATISTICS ===
    public List<ListenLog> getRecentListenLogs() {
        return listenLogRepository.findTop50ByOrderByTimestampDesc();
    }
```

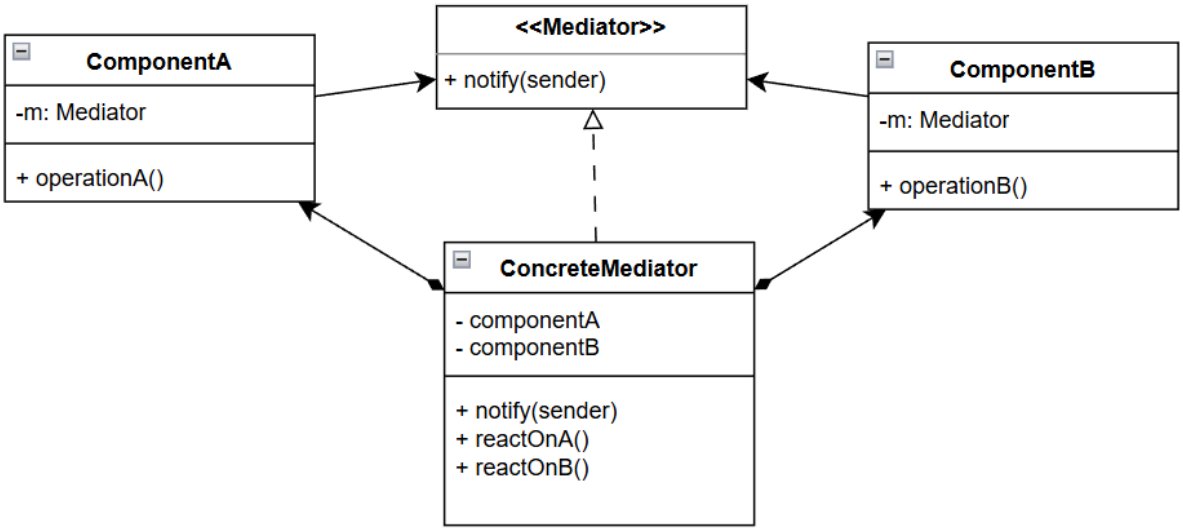
```
    public List<ConnectionLog> getRecentConnectionLogs() {
        return connectionLogRepository.findTop50ByOrderByTimestampDesc();
    }

}
```

Контрольні питання:

1. Яке призначення шаблону «Посередник»?

Визначає об'єкт, який інкапсулює спосіб взаємодії множини об'єктів. Що дозволяє уникнути явних посилань об'єктів один на одного

2. Нарисуйте структуру шаблону «Посередник».



3. Які класи входять в шаблон «Посередник», та яка між ними взаємодія?

Mediator: Інтерфейс взаємодії.

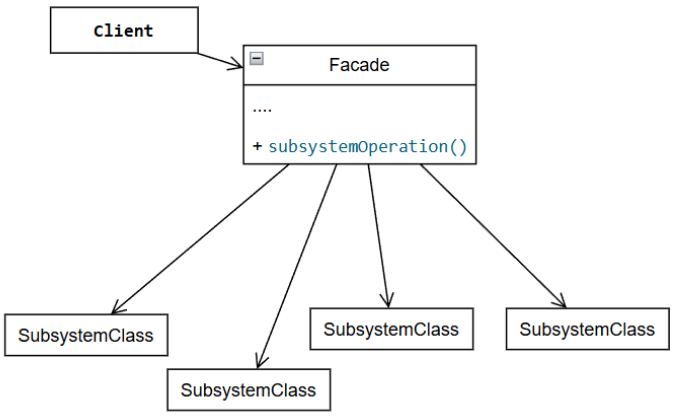
ConcreteMediator: Знає про колег і координує їх.

Colleague (Component): Взаємодіє лише з посередником.

4. Яке призначення шаблону «Фасад»?

Передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття її внутрішніх деталей

5. Нарисуйте структуру шаблону «Фасад».



6. Які класи входять в шаблон «Фасад», та яка між ними взаємодія?

Є клас Facade, який знає, яким класам підсистеми переадресувати запит.

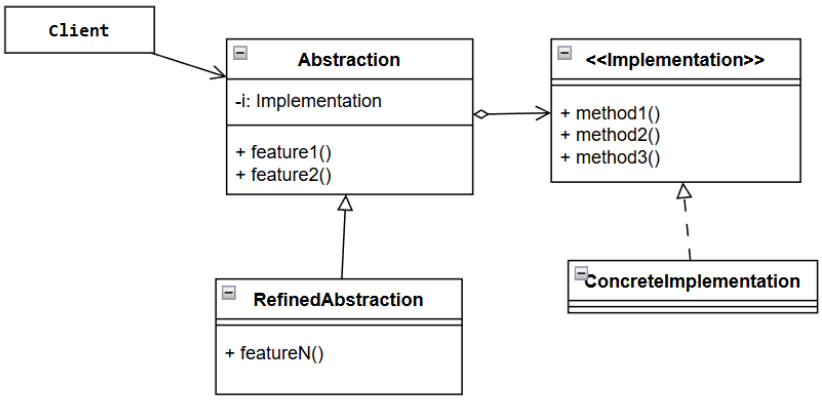
Є класи Subsystem, які виконують фактичну роботу.

Клієнти взаємодіють з Фасадом. Фасад перенаправляє виклики відповідним об'єктам у підсистемі. Класи підсистеми не знають про Фасад.

7. Яке призначення шаблону «Міст»?

Використовується для поділу абстракції (інтерфейсу) та її реалізації, щоб вони могли змінюватися незалежно одна від одної.

8. Нарисуйте структуру шаблону «Міст».



9. Які класи входять в шаблон «Міст», та яка між ними взаємодія?

Abstraction: Визначає інтерфейс абстракції та зберігає посилання на об'єкт Implementation.

RefinedAbstraction: Розширює інтерфейс абстракції.

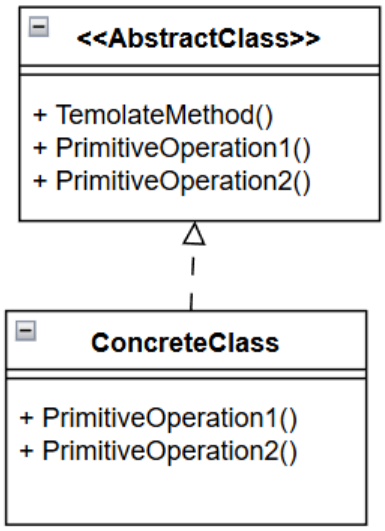
Implementation: Визначає інтерфейс для класів реалізації.

Concrete Implementation: Конкретна реалізація інтерфейсу Implementation.

10. Яке призначення шаблону «Шаблоний метод»?

Визначає скелет алгоритму в базовому класі, але залишає реалізацію деяких кроків підкласам. Дозволяє перевизначати частини алгоритму без зміни його структури.

11. Нарисуйте структуру шаблону «Шаблоний метод».



12. Які класи входять в шаблон «Шаблоний метод», та яка між ними

взаємодія?

AbstractClass: Реалізує шаблонний метод, який викликає примітивні операції.

ConcreteClass: Реалізує примітивні операції.

Клієнт викликає шаблонний метод. Цей метод виконує кроки алгоритму, викликаючи абстрактні методи, які реалізовані в конкретному підкласі

13. Чим відрізняється шаблон «Шаблонний метод» від «Фабричного

методу»?

Головна відмінність полягає у призначенні: шаблонний метод – це поведінковий патерн, який визначає загальний скелет алгоритму, дозволяючи підкласам змінювати реалізацію окремих його кроків без порушення структури. Фабричний метод – це породжувальний патерн, який фокусується виключно на одному кроці – створенні об'єкта, делегуючи це рішення підкласам.

14. Яку функціональність додає шаблон «Міст»?

Шаблон міст додає функціональність відокремлення абстракції від її реалізації, дозволяючи їм змінюватися незалежно одна від одної. Це запобігає вибуховому зростанню кількості класів при масштабуванні системи в кількох вимірах та дозволяє підміняти реалізацію динамічно під час виконання програми.

Висновок: Під час виконання даної лабораторної роботи було досягнуто поставлену мету: вивчено та проаналізовано групу шаблонів проєктування, що використовуються для вирішення типових задач в об'єктно-орієнтованому програмуванні. Було детально розглянуто призначення, структуру та принципи взаємодії класів для наступних патернів: «Mediator», «Facade», «Bridge», «Template method». А також на практиці в програмному коді було реалізовано шаблон «Фасад».